AMATH 3132: Numerical analysis I

Jahrul Alam

Department of Mathematics and Statistics
Memorial
University of Newfoundland

Winter 2010

Introduction to matlab

- MATLAB stands for MATrix LABoratory.
- Matlab is developed by The Mathworks Inc (www.mathworks.com)
- High level programming language.
- Matlab runs as interpreted mode as opposed to compiled mode.
- Matlab is an integrated environment for
  - numerical computations.
  - scientific visualizations.

Introduction to matlab

- Slow compared with FORTRAN or C.
- Automatic memory management.
- Easy to use.
- Shorter program development time.
- Can be converted to C.
- Many tool boxes are available.
- Certain operations can be processed in parallel.

Introduction to matlab

- Invoke by typing **matlab** in the system command prompt.
- Matlab desktop appears with a command prompt "≫".
- type `exit` or `quit` to exit matlab.
- HELP
  - '%' is the symbol for comments.
  - ≫`help` % prints list of available packages.
  - ≫`help package` % prints available functions in the package.
  - ≫`help function` % prints instruction of the function.

Introduction to matlab

- `x = 1;` % assigns 1 to the variable *x*.
- `x = 1:8;` % assigns an array to *x*.
- `size(x)` % prints the size [1 8] of *x*.
- `length(x)` % prints length 8 of *x*.
- Try `x=1:2:8`
- `clear x` % clears assigned values of *x*.
- `clear all` % clears all variable in the work space.

Introduction to matlab

- `abs(x)` % absolute value of x
- `exp(x)` % e to the x-th power
- `fix(x)` % rounds x to integer towards 0
- `log10(x)` % common logarithm of x to the base 10
- `rem(x,y)` % remainder of x/y
- `sqrt(x)` % square root of x
- `sin(x)` % sine of x; x in radians
- `acoth(x)` % inversion hyperbolic cotangent of x
- `help elfun` % get a list of all available elementary functions

Script file

- Create a file test.m with an extension .m. Append the following code, and save.

```
disp('Hello World');
x=30*pi/180;
a = sin(x);
disp(a);
```

- Use matlab command ≫`test`;
- Above four line codes are executed.

Matlab function m-file

- We want a function to invoke previous four lines code.
- Create a file test.m with an extension .m. Append the following code, and save.

```
function test()
disp('Hello World');
x=30*pi/180;
a = sin(x);
disp(a);
```

- Use matlab command ≫`test()`;
- Above four line codes are executed.
- Function name and file name should be the same.
- There is no end function statement.

Matrix operations

- Let us create a 4×4 matrix.
```
for i=1:4
for j=1:4
a(i,j)=i*j;
end
end
```
- Can find eigenvalues
```
eig(a)
```
- Can invert the matrix:
```
inv(a)
```
- Singular value decomposition:
```
svd(a)
```
- Transpose
```
a'
```


Matrix operations

- Enter a row vector
```
≫x=[1 2 3];
```
- Enter a column vector
```
≫x=[1; 2; 3];
```
- Enter a 3×3 matrix
```
≫A=[1 2 3;4 5 6;7 8 9];
```
- Special matrices:
  - `≫Z=zeros(4,6);`
  - `≫O=ones(4,6);`
  - `≫I=eye(4,6);`
  - `≫D=diag([1 2 3 4]);`

Matrix operations

- `≫C=A+B;`
- `≫C=A-B;`
- `≫C=A.^2;`
- `≫C=A.*3;`
- `≫A=[1 2; 3 4]`
```
ans=

1 2
3 4
```
- `≫A.^2`
```
ans=

1 4
9 16
```


Programming

- Operators

==	Equal
~=	Not equal
<	Less than
>	More than
<=	Less than or equal
>=	More than or equal
~	NOT
&	AND
|	OR
any	True if any element is nonzero
all	True if all elements are nonzero

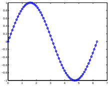
Programming

Control statements

- IF
```
if (logical expression)
...
elseif (logical expression)
...
else
...
end
```
- WHILE
```
while(while expression)
...
end
```

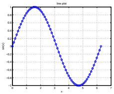

Graphics

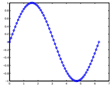
- Line plot
```
x=linspace(0,2*pi,64);
y=sin(x);
plot(x,y,'o-');
```



Graphics

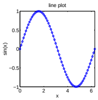
- Line plot
```
x=linspace(0,2*pi,64);
y=sin(x);
plot(x,y,'o-');
```
-
```
xlabel('x');
ylabel('sin(x)');
title('Line plot');
grid on;
```

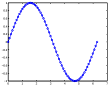




Graphics

- Line plot
```
x=linspace(0,2*pi,64);
y=sin(x);
plot(x,y,'o-');
```
-
```
xlabel('x');
ylabel('sin(x)');
title('Line plot');
grid on;
```
-
```
set(gca,'fontsize',18);
axis([0 2*pi -1 1]);
axis square;
grid off;
```

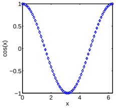


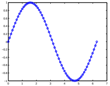


Graphics

- Two plots in one frame.

```
z=cos(x);
plot(x,z,‘d--’);
ylabel(‘cos(x)’);
grid off;
```





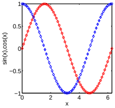
Graphics

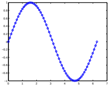
- Two plots in one frame.

```
z=cos(x);
plot(x,z,‘d--’);
ylabel(‘cos(x)’);
grid off;
```

- hold on;

```
plot(x,y,‘ro-’);
```





Graphics

- Two plots in one frame.

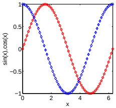
```
z=cos(x);
plot(x,z,‘d--’);
ylabel(‘cos(x)’);
grid off;
```

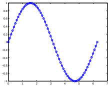
- hold on;

```
plot(x,y,‘ro-’);
```

- clf;

```
plot(x,y,‘ro-’,x,z,‘d--’);
xlabel(‘x’);
ylabel(‘sin(x),cos(x)’);
```

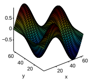




Graphics

- Surfaces

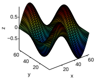
```
x=linspace(0,2*pi,32);
[x,y]=meshgrid(x,y);
z=sin(x) .* cos(x);
surf(x,y,z);
set(gca,‘fontsize’,18);
xlabel(‘x’);
ylabel(‘y’);
zlabel(‘z’);
axis square;
grid off;
```



Graphics

- Surfaces

```
x=linspace(0,2*pi,32);
[x,y]=meshgrid(x,y);
z=sin(x) .* cos(x);
surf(x,y,z);
set(gca,'fontsize',18);
xlabel('x');
ylabel('y');
zlabel('z');
axis square;
grid off;
shading interp;
```



Error

- What is error?
  In scientific computing, we often need to approximate some functions or solutions of equations. This results into a defect between the true value and the approximated value. This defect is called error.
- Sources of error.
  - **Truncation error**
  - **Round-off error**
  - **Propagated error**

Error

- What is **truncation error**?
  When a function can be approximated by an infinite series, but the series is truncated up to a finite number of terms, the discarded terms introduce an error that is known as truncation error.
- Let
  $$p_4(x) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!}$$
  be a 3-rd degree polynomial. We know that
  $$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots\ldots$$

Error

- If we approximate e^x by $p_4(x)$ such that $e^x \approx p_4(x)$ then the truncation error is
  $$e^x - p_4(x) = \sum_{n=4}^{\infty} \frac{x^n}{n!}.$$
- A trunction error is made by the numerical approximation of continuous functions.
- What is **round-off error**?
  Numerical error also occurs due to rounding floating point numbers.

Error

- What is **propagated error**?
  Often we need to calculate a solution sequentially or successively, where a calculation uses previously calculated values.
- Therefore, in addition to error made in the present calculation i.e. output, error from the previous calculation i.e. from input is also added to the output.
- This type of error is called propagation error. Such error can accumulate continuously if the calculation is repeated many times.

Error

- If the propagated error accumulates excessively in an algorithm, the validity of the output is often destroyed. Such algorithms or methods are unstable.
- If the accumulation of the propagated error does not grow rapidly with the number if repeatation, we call the method stable.
- Note that a stable method is not free from propagated error.

Size of error

- **Absolute error**: The magnitude of the difference between the exact value and the approximate value is called absolute error, and is defined by
  absolute error = | exact value - approximate value |.
- **Relative error**: It is often useful to measure error relative to the exact value such that
  relative error $= \frac{\text{absolute error}}{|\text{exact value}|}$

Size of error

- **Example**: Exact value $x_e = 10/3$ and approximate value $x_a = 3.333$. Then

$$\text{absolute error} = |\ 10/3 - 3.333\ | = 0.000333\ldots.$$

$$\text{relative error} = \frac{0.000333\ldots}{10/3} = 0.0000999\ldots$$

Floating point number system

A floating point number system is characterized by four integers:

β	Base
p	Precision
$[L, U]$	Exponent range

Any floating point number has the form:
$x = \pm\left(d_0 + \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \dots \frac{d_{p-1}}{\beta^{p-1}}\right)\beta^E$, where d_i is an integer such that $0 \le d_i \le \beta - 1$ for $i = 0 \dots p-1$, and E is an integer such that $L \le E \le U$.

- Mantissa: a string of p base-β digits $d_0 d_1 \dots d_{p-1}$.
- Exponent: E is the exponent.
- Fraction: the portion $d_1 \dots d_{p-1}$ of the mantissa.

In a computer, the sign, exponent, and mantissa are stored in sperate fields of a given floating point word.

Floating point number system

- Any positive real number within the numerical range of the machine is expressed in the normalized form.

$$x = 0.d_1 d_2 \dots d_k \dots \times 10^n,$$

where $1 \le d_1 \le 9$ and $0 \le d_i \le 9$ for each $i = 2 \dots k$.

- **Chopping**: We chop off all digits $d_{k+1} \dots$ and write

$$x = 0.d_1 d_2 \dots d_k \times 10^n.$$

- **Rounding**: We add $5 \times 10^{n-(k+1)}$ to x and then chop off to get

$$x = 0.\delta_1 \delta_2 \dots \delta_k \times 10^n.$$

Floating point number system

- Example

$$\pi = 0.314159265 \dots \times 10^1,$$

which is written as $\pi = 3.1415$ using a five digit chopping, or as $\pi = 3.1416$ using a five-digit rounding.
- Note that we add 5×10^{-5} to π and then chop off to round the number.

Data representation in a computer

- A decimal integer I can be represented in a binary form

$$(I)_{10} = (a_{m-1} \dots a_2 a_1)_2 = \sum_{j=0}^{m-1} 2^j (a_j).$$

- The procedure can be described as:

$$\begin{aligned} I &= 2P_0 + a_0 \\ P_0 &= 2P_1 + a_1 \\ P_1 &= 2P_2 + a_2 \\ &\vdots \\ P_{m-2} &= 2P_{m-1} + a_{m-1} \end{aligned}$$

The process is stopped if $P_{m-1} = 0$.

Data representation in a computer

- A fractional number R, $0 < R < 1$ can be represented as

$$\begin{aligned} R &= b_1 \times \beta^{-1} + b_2 \times \beta^{-2} + \cdots + b_m \times \beta^{-m} \\ &= 0.b_1 b_2 \ldots b_m \end{aligned}$$

- If $\beta = 10$ and $0 \leq b_i \leq 9$, we get R in the decimal system.
- If $\beta = 2$ and $0 \leq b_i \leq 1$, we get R in the binary system.

Data representation in a computer

We can convert a real number $R, 0 < R < 1$ to a binary number. The procedure is described below:

$$\begin{aligned} 2R &= b_1 + f_1 \\ 2f_1 &= b_2 + f_2 \\ &\vdots \\ 2f_{m-1} &= b_m + f_m \end{aligned}$$

where b_m=int($2f_{m-1}$) and f_m=fraction($2f_{m-1}$).
The process is continued until the fractional part $f_m = 0$.

Errors associated with arithmetic operations

- Numbers cannot be stored exactly by the floating point representation. Let x and y be the exact numbers and $\tilde{x}$ and $\tilde{y}$ their approximate values. Then,

$$x = \tilde{x} + \epsilon_x, \; y = \tilde{y} + \epsilon_y,$$

  where ϵ_x and ϵ_y denote the errors in x and y respectively.
- Find the error associated with a multiplication operation.
- $E = xy - \tilde{x}\tilde{y} = xy - (x - \epsilon_x)(y - \epsilon_y) = x\epsilon_y + y\epsilon_x - \epsilon_x\epsilon_y$.

Errors associated with arithmetic operations

- There relative error

$$\begin{aligned} R &= \frac{R}{xy} = \frac{\epsilon_x}{x} + \frac{\epsilon_y}{y} - \frac{\epsilon_x}{x}\frac{\epsilon_y}{y} \\ &= R_x + R_y - R_x R_y \\ &\approx R_x + R_y, \end{aligned}$$

  where $| R_x | \ll 1$ and $| R_y | \ll 1$
  Example:
- Let $x = 2.71828183$, $y = 2.71818283$ and $\tilde{x} = 2.7183$, $\tilde{y} = 2.7181$. Determine the error and relative error associated with the subtraction $x - y$.

Errors associated with arithmetic operations

- Solution:

$$E_x = x - \tilde{x} = -1.917 \times 10^{-5},\ R_x = -6.684 \times 10^{-6}$$

$$E_y = y - \tilde{y} = 8.283 \times 10^{-5},\ R_y = 3.0473 \times 10^{-6}$$

- The error associated with subtraction:

$$E = (x - y) - (\tilde{x} - \tilde{y}) = E_x - E_y = -10.1 \times 10^{-5}$$

$$R = \frac{E}{x - y} = \frac{-10.1 \times^{-5}}{0.000099} \approx -0.10202020.$$

- We see that E is small, but R is large.

Errors associated with arithmetic operations

- Find the roots of $x^2 + 62.10x + 1 = 0$ with the formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

using a four-digit rounding arithmetic.

- Suppose that we know the roots:

$$x_1 = -0.01610723,\ x_2 = -62.08390.$$

- Calculate the relative error if arithmetic operations occur between nearly equal number.
- Propose and justify an improvement.

Errors associated with arithmetic operations

- Solution: Note that

$$\sqrt{b^2 - 4ac} = \sqrt{(62.10)^2 - (4.000)(1.000)(1.000)} = 62.06.$$

- We now get

$$x_1 = \frac{-62.10 + 62.06}{2.000} = \frac{-0.04000}{2.000} = -0.02000.$$

Note the subtraction between nearly equal numbers!

- The relative error

$$R_{x_1} = \frac{|-0.01611 + 0.02000|}{|-0.01611|} \approx 2.4 \times 10^{-1}.$$

relative error is large!

Errors associated with arithmetic operations

- To avoid subtraction between nearly equal numbers, consider

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}\left(\frac{-b - \sqrt{b^2 - 4ac}}{-b - \sqrt{b^2 - 4ac}}\right),$$

which implies to

$$x_1 = \frac{-2c}{b + \sqrt{b^2 - 4ac}}.$$

- We now get

$$x_1 = \frac{-2.000}{62.10 + 62.06} = -0.01610.$$

- The relative error $= 6.2 \times 10^{-4}$